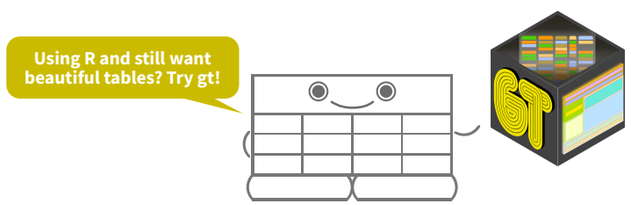
Great Tables : : CHEAT SHEET

Package Aims

Great Tables allows you to make beautiful and presentable table displays from Python DataFrames (Polars, Pandas, and Arrow). The package provides methods for you to **structure**, **format**, and **style** the table to your heart's content.

The resulting tables can be easily displayed in notebooks and Quarto documents. And there are options for saving images of a table or getting HTML or LaTeX table code.

INSTALLATION `pip install great_tables`



Adding Structure

ADDING A HEADER AND FOOTER

`tab_header()` → add title and optional subtitle, can use `md()` or `html()` helpers

`tab_source_note()` → add text to table footer (helpers can be used here too)

ADDING A STUB AND ROW GROUPS

`GT(rowname_col="column")` → assign a column with row names to the stub

`GT(groupname_col="column")` → column with categoricals groups rows

ADDING COLUMN SPANNERS

`tab_spanner()` → add spanner over cols

`tab_spanner_delim()` → use delimited col names to set up multiple spanners

Styling the Table

```
from great_tables import loc, style
```

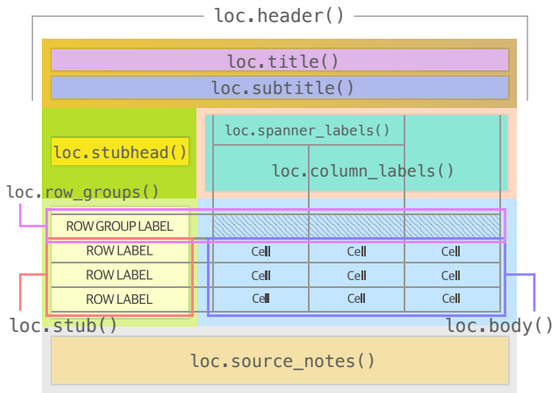
import these two modules

ONE METHOD FOR STYLING EVERYWHERE

`tab_style()` → apply styles to various locations

```
gt_tbl.tab_style(style=[styles], locations=[locations])
```

THE DIFFERENT LOCATIONS OF A TABLE



THREE WAYS TO STYLE

`style.fill()` → fill location with color

`style.text()` → style location's text

`style.borders()` → set up borders and define styles for them

YOU ALSO HAVE OTHER OPTIONS

`tab_options()` → many, many options for styling that applies to whole table

`opt_stylize()` → set a theme, 36 looks

`opt_table_font()` → set a font to use

Creating a GT Table

BASIC EXAMPLE WITH SAMPLE DATA

import GT class

```
from great_tables import GT
from great_tables.data import islands
```

dataset

```
gt_tbl = GT(islands.head(5))
gt_tbl
```

create GT object

display HTML table

OUTPUT

column labels

name	size
Africa	11506
Antarctica	5500
Asia	16988
Australia	2968
Axel Heiberg	16

table body

SAMPLE DATASETS

`countrypops`, `sza`, `gtcars`, `sp500`, `pizzaplace`, `exibble`, `towny` + more

SHOWING IN BROWSER

```
gt_tbl.show("browser")
```

POLARS STYLE PROPERTY

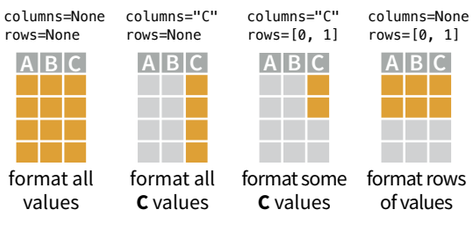
```
df_polars.style
```

create GT object with Polars API

Formatting Data Values

FORMATTING BASICS

Formatting methods have the `columns` and `rows` parameters to specify which body values should be formatted.



NUMERIC FORMATTING

1 `fmt_number()` → decimal values
2 `fmt_integer()` → integer values
3 `fmt_percent()` → percentage values
4 `fmt_scientific()` → scientific not'n

common params: `decimals` in 1,3,4 for fixed # of decimals / `scale_by` in 1,2,4 to multiply number before formatting / `accounting` in 1,2,3 to display in that notation / `compact` in 1,2 to condense large numbers / `locale` in 1–4 to localize formatted values to language/region.

DATE/TIME FORMATTING

`fmt_date()` → format with a date style

```
iso: "2000-02-29" (the default)
wday_month_day_year: "Tuesday, February 29, 2000"
wd_m_day_year: "Tue, Feb 29, 2000"
wday_day_month_year: "Tuesday 29 February 2000"
month_day_year: "February 29, 2000"
```

`fmt_time()` → format with a time style

```
iso: "14:35:00" (the default)
h_m_s_p: "2:35:00 PM"
h_m_p: "2:35 PM"
```

`fmt_datetime()` → format with date/time styles or with the `format_str` parameter

FORMATTING INVOLVING GRAPHICS

`fmt_image()` → insert images to cells with filenames; `path` / `file_pattern` help resolve

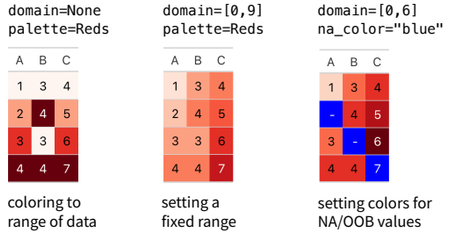
`fmt_icon()` → add icons to cells containing FontAwesome short icon names (e.g., `"cat"`)

`fmt_flag()` → add flag icons to cells w/ 2- or 3-letter country codes (e.g., `"LU"` or `"LUX"`)

common to all: cells may have multiple instances of replaceable text for these formatters, they must be comma-separated though.

COLORIZING BODY CELLS

`data_color()` → apply color to a series of cells according to their value



ADDING NANOPLOTS

`fmt_nanoplot()` → convert series of values (strings w/ vals or a list column) to tiny interactive plots

	vals
1	20 23 6 7 37 23 21 4 7 16
2	2.3 6.8 9.2 2.42 3.5 12.1 5.3 3.6